**Titanic\_dataset**

**Problem Definition**

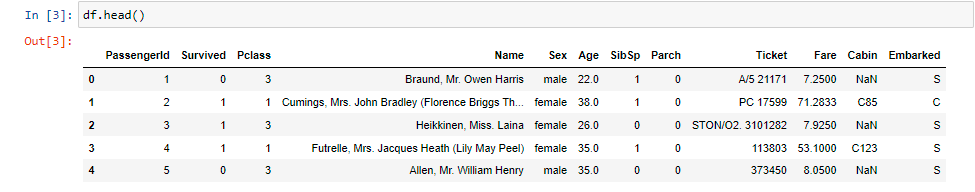
The Titanic Problem is based on the sinking of the ‘Unsinkable’ ship Titanic in early 1912. It gives you information about multiple people like their ages, sexes, sibling counts, embankment points, and whether or not they survived the disaster. Based on these features, you have to predict if an arbitrary passenger on Titanic would survive the sinking or not.

**1).Data Acquistion -** In data acquisition, we collect data from certain websites, like, kaggle and UCI that becomes the training data and is useful to make predictions. But, when we acquire data then we should be sure enough that data must have enough features so that we can make predictions easily. Generally, data must be in the form of CSV, i.e. , comma separated value and currently the most supported size is 1.95 but it is not appropriate for big data.

In this project, we have loaded titanic dataset in the form of CSV by using kaggle with the help of pandas, which is the most popular python library that is used for data analysis, and it mainly works on dataset, like, it is used for merging, joining, reshaping, pivoting, aligning, analyzing and arranging datasets. We have imported numpy which is mainly used for calculations in our dataset in the following commands :-



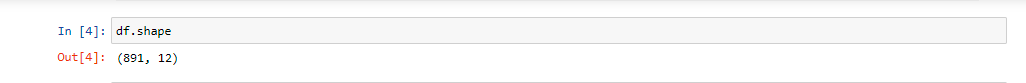
And following is the illustration of dataset in which by using **df.head()** from whichwe can see the first five rows of the dataset:-



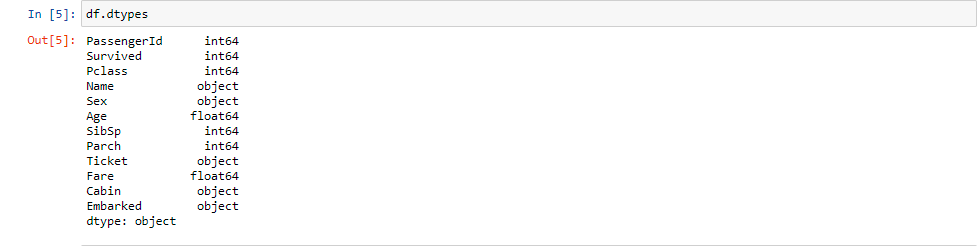
* **PassengerId** - shows an unique index for passenger rows that starts from 1 for first row and increases by 1 for every new row.
* **Survived** - reveals that if the passenger survived or not, where, 1 stands for survived and 0 stands for not survived.
* **Pclass** - 1 stands for first class ticket, 2 stands for second class ticket and 3 stands for third class ticket.
* **Name** - shows the name of the passenger.
* **Sex** - represents the gender of the passenger.
* **Age** - represents the age of the passenger.
* **SibSp**- shows the number of siblings or spouses travelling with each passenger.
* **Parch** - represents the number of parents or children travelling with each passenger.
* **Ticket**- represents the ticket number of each passenger.
* **Fare**- represents the fare of each passenger.
* **Cabin** - represents the cabin number of each passenger .
* **Embarked** - shows the port from where the particular passenger was embarked or boarded.

**2).Data Cleaning -** Due the problem of incorrect or inconsistent data in both public and private sector which led to fast inferences and investments, there is a need of some packages which can clean or wash our address data when we enter them into our system using application programming interface which is a set of subroutine definitions, communication protocols and tools for building software. Data cleaning is the process of detecting and correcting corrupt or inaccurate records from a record set, table or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty data. Data is also missing because sometimes user don’t remember to fill it in a field, sometimes data is gone while shifting it manually from a provision database and sometimes a programming error has occurred. Mostly, 80 % of work is done in this step of data cleaning.

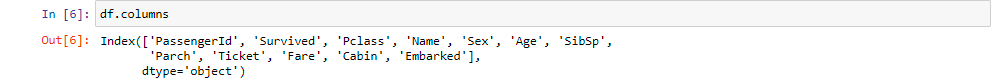
* We can check the shape of the dataset by using the code- **df.shape()** from which we can observe that there are 58 rows and 18 columns in this dataset using the following command:-



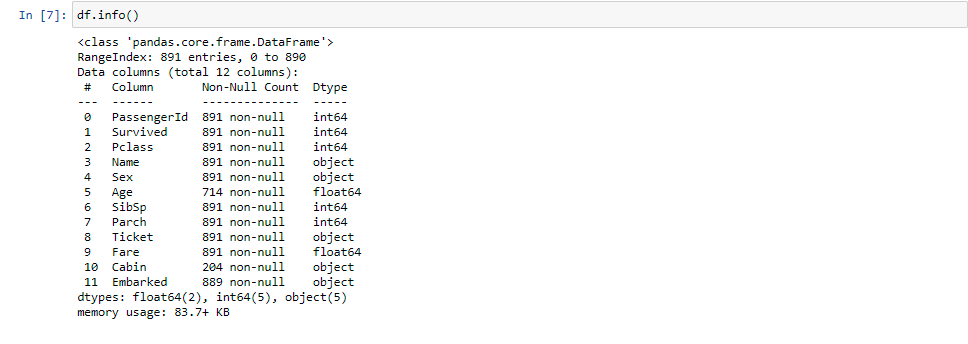
* We can check the datatypes of the columns by using-**df.dtypes**. From the following commands we can observe that there are four categorical feature present in this dataset Name’, 'Sex', 'Ticket', 'Cabin', ‘Embarked’ and rest are Numerical features: ‘Passenger ID’, ‘Survived’, ‘Pclass’, ‘Age’, ‘SibSp’, ‘Parch’ and ‘Fare’.



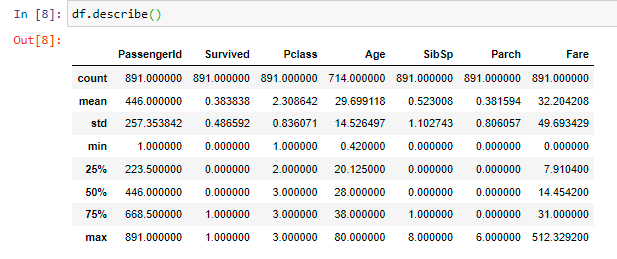
* We can check the columns present in this dataset by using the code-**df.columns** in the following command :-



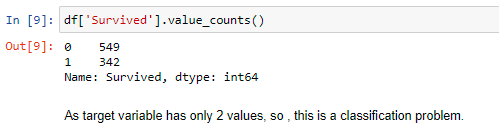
* After loading the file we can start analyzing our dataset such as checking the information about dataset using **df.info()** in the following command :-



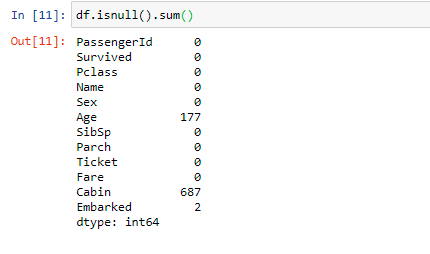
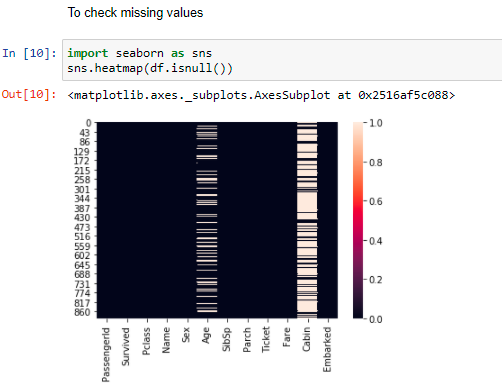
* **Summary Statistics**
* Now, using the following command of data.describe ( ) we find count, mean, standard deviation, minimum, maximum, 25%, 50% and also 75% of our dataset which is used to decide that which column contribute more towards training of our model to make certain predictions and also used in testing of our model in the following commands :-



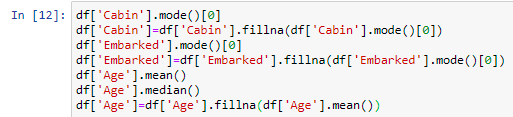
* From this we can observe that :-
* The mean is less than median for all columns.
* There is large difference between 75 % and maximum for fare and Passenger Id column.
* In the following command, we calculate count different types of values in Survived column using **df['Survived'].value\_counts()** and as target variable has only 2 values, so, this is a classification problem:-



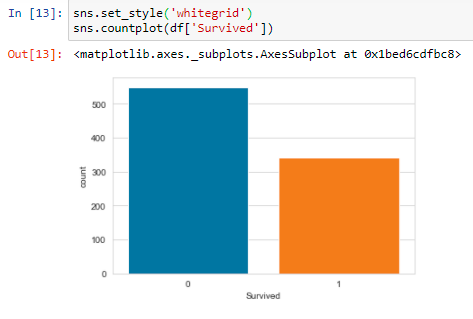
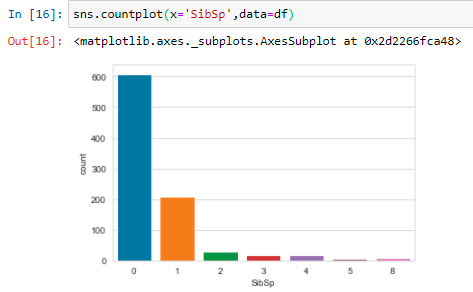
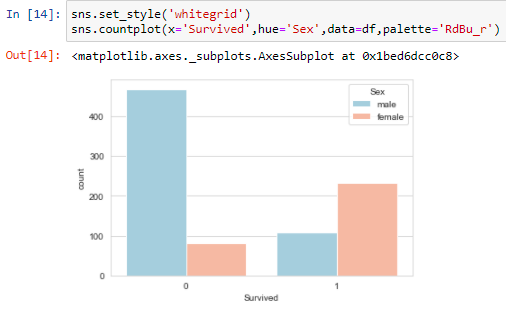
* **Data Visualizations**
* In order to clean our data, we have to clean it by checking that if there are null values in our data or not by using the command- **df.isnull().sum()** and if there are null values in our data then we replace these null values with some objects. Similarly, we can visualize it with the help of heatmap by using the following command :-

****

* From this, we can observe that missing values are present in ‘Age’, ‘Cabin’ and ‘Embarked’ columns.
* **Handling Missing Values**
* So we will impute(replace) the missing values of categorical columns with mode of the corresponding columns and the missing values of categorical column with the mean of the corresponding column as follows :-

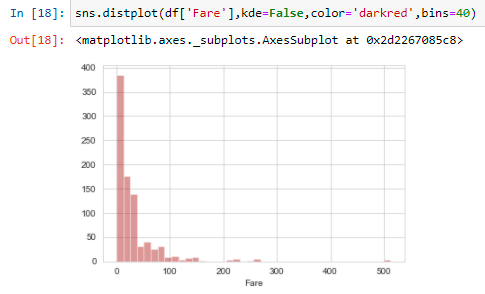
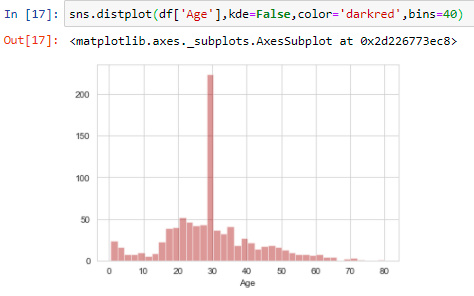
****

* **Univariate Analysis**
* In this section, we analyze attributes individually; check their distribution and range of values. We use Countplot to analyze ‘Survived’ attribute individually and analyze ‘Survived’ attribute in relation with 'Sex', 'Pclass' and 'SibSp' attributes. Since these attributes have categorical values and can be better visualized using countplot by the following commands :-

From these it can be observed that:-

* Passengers have sinked more than survived.
* Females have survived more than males.
* Maximum number of passengers have survived in Passenger Class 1.
* Minimum number of passenger have survived in Passenger Class 3.
* 600 number of passenger have neither sibling nor spouse with them.
* 5 number of passenger have 5 sibling or spouse with them.
* We use distplot to analyze 'Age' and 'Fare' attributes individually. Since these attributes have numerical values and can be better visualized using distplot by the following commands :-



From these it can be observed that :-

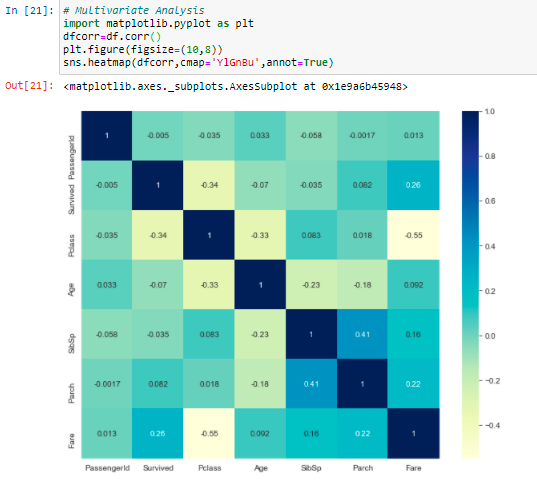
* Maximum number of passenger have survived between the age of 20 to 40.
* Least number of passenger have survived between the age of 60 to 90.
* Maximum number of passengers have paid fare between 0 to 100.
* Few passengers have paid fare between 200 to 300.
* No passenger has paid fare more than 300.
* **Bivariate Analysis**
* In this section, we will be analyzing the impact of each attribute on each other. We use violinplot to analyze the impact of 'Age' on ‘Sex’ and 'Pclass' attributes. Since ‘Sex’ and ‘Pclass’ attributes have categorical values and 'Age' has numerical values. So, It can be better visualized using violinplot by the following commands :-





From these it can be observed that:-

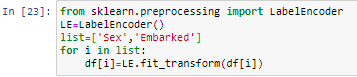
* For the age range of 20-40, in class 1, Passengers have survived more than died.
* For the age range of 20-40, in class 2, Passengers have died more than survived.
* For the age range of 20-40, in class 3, Passengers have died more than survived.
* For the age range of 0-20, in class 1, Passengers have survived more than died.
* For the age range of 0-20, in class 2, Passengers have survived more than died.
* For the age range of 0-20, in class 3, Passengers have survived more than died.
* For the age range of 40-60, in class 1, Passengers have died more than survived.
* For the age range of 40-60, in class 2, Passengers have died more than survived.
* For the age range of 40-60, in class 3, Passengers have died more than survived.
* For the age range of 20-40, in male category, Passengers have died more than survived.
* For the age range of 20-40, in female category, Passengers have survived more than died.
* For the age range of 0-20, in male category, Passengers have survived more than died.
* For the age range of 0-20, in female category, Passengers have died more than survived.
* For the age range of 40-60, in male category, Passengers have survived more than died.
* For the age range of 40-60, in female category, Passengers have died more than survived.
* **Multivariate Analysis**
* We can check the correlation of all the attributes with the target variable-Deaths as follows:

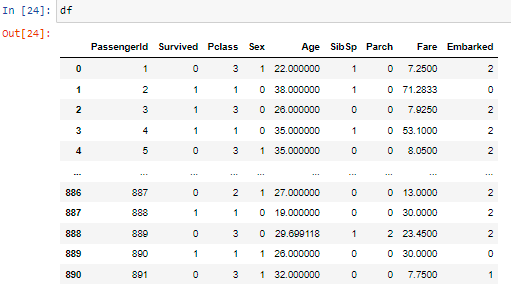


* From these it can be observed that :-
* Survived is highly positively correlated with Fare.
* Fare is highly negatively correlated with PClass.
* PClass is highly positively correlated with Embarked.
* PClass is highly negatively correlated with Survived.
* **Dropping of Columns**
* We drop Name, Ticket and Cabin columns as they have no impact on target variable.

****

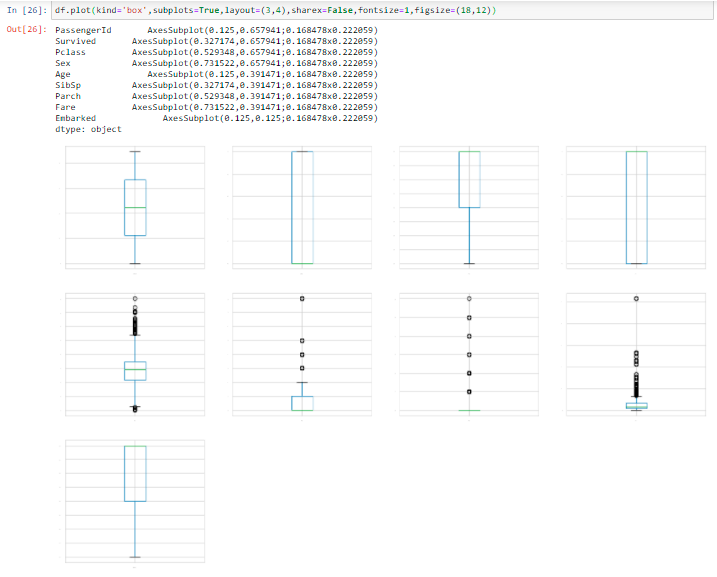
* **Label Encoding**
* We encode all the categorical variables to numerical ones by using “Label Encoder” as shown below:

****

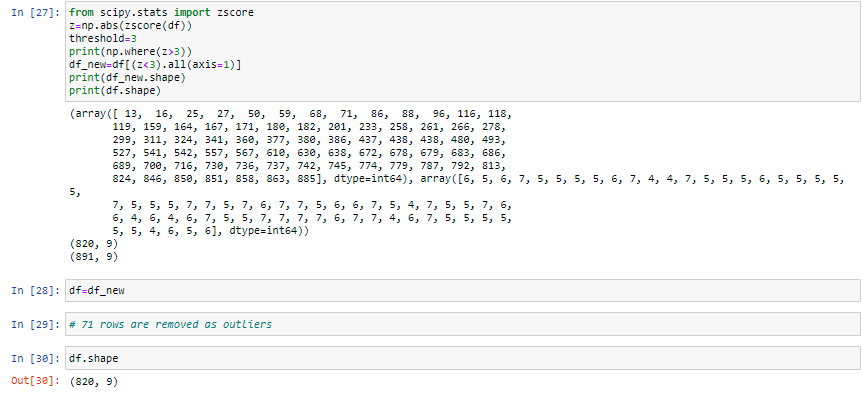
****

****

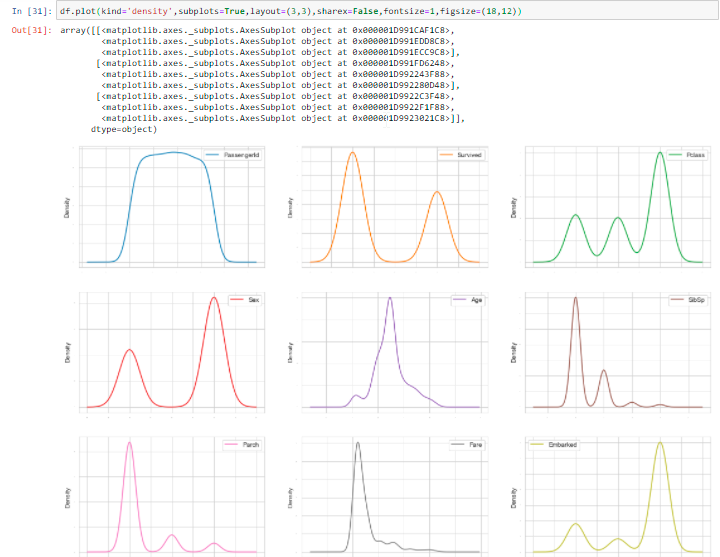
* The final shape of our dataset after dropping 3 columns are (891,9).
* **Plotting Outliers**
* Now we will be plot the outliers by using boxplot for all the attributes as follows:

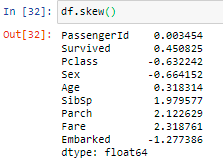


* From these it can be observed that :-
* Age, SibSp, Parch, Fare and Embarked have outliers.
* **Removing Outliers**
* In order to remove outliers from our data, we have to remove outliers by using z-score or IQR method. Here we are using z-score method as following :-

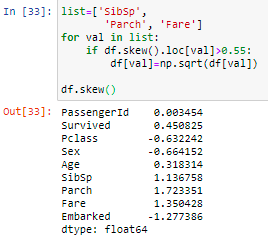
****

* We have removed 71 rows from the data as outliers were present in them. Now, the dataframe has 820 rows and 9 columns.
* **Removing Skewness**
* In order to remove skewness from our data, we have to remove skewness by checking that whether the attributes are left skewed or right skewed using- **df.skew()**. Similarly, we can visualize it with the help of distplot by using the following command :-

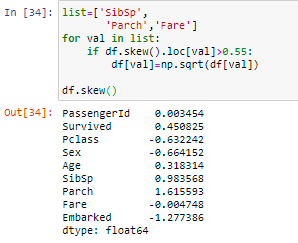
****

****

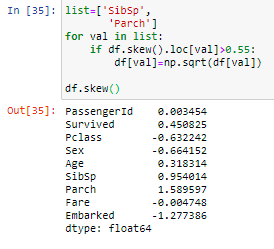
* From these it can be observed that :-
* SibSp, Parch and Fare are right skewed.
* Sex, Embarked and Pclass are left skewed.
* We can treat the attributes that are having skewness above 0.55 by any method. Here we are using log transform to remove skewness in the following commands :-



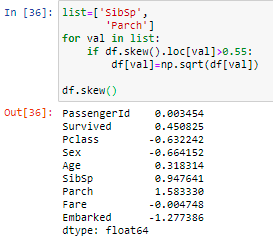
* After applying log transformation method, following are the observations :-
* Skewness of SibSp, Parch and Fare have reduced to some extent but it has not removed completely.
* We apply square root transformation method in the following commands :-



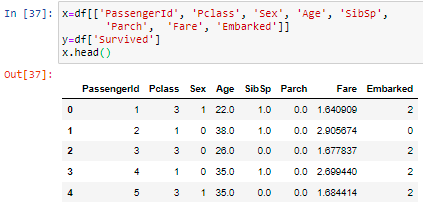
* After applying sqrt transformation method, following are the observations :-
* Skewness of SibSp and Parch have reduced to some extent but skewness of Fare has removed completely.
* We apply square root transformation method in the following commands :-



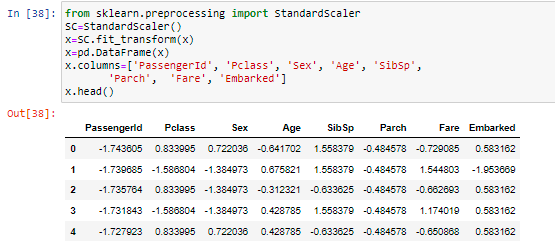
* After applying sqrt transformation method, following are the observations :-
* Skewness of SibSp and Parch have reduced to some extent but it has not removed completely.
* We apply square root transformation method in the following commands :-



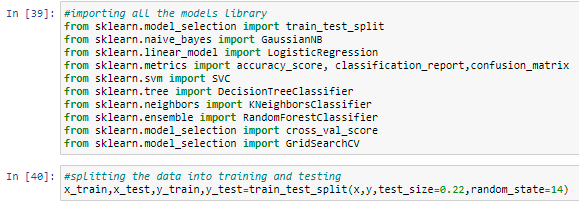
* After applying sqrt transformation method, following are the observations :-
* Skewness of SibSp and Parch have reduced to some extent but it has not removed completely.
* We do not apply any transformation because by applying the transformation methods on same attributes more than three or four times can create null values in the attributes. Thus, we stop applying transformation methods now to avoid null values.
* **Train Test Split**
* We usually split our data into training data which contains a known output so that a model learn on this data in order to be generalized to other data and testing data in order to test our model on subset for prediction. Training data must contains those variables which affect testing data and it should be also clean, properly classified, not containing nan values and should contain more numerical data rather than categorical data. Training and testing data must be divided in the ratio of 80:20. In this project, we have chosen our train and target columns as following to make predictions about the “Survived” column :-

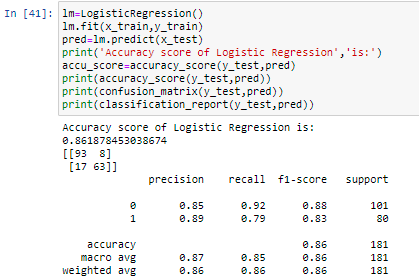
****

* **Scaling of Data**
* In order to scale our data, we have to scale data by using Standard Scaler or Min-Max Scaler method. Here we are using Standard Scaler method as following :-

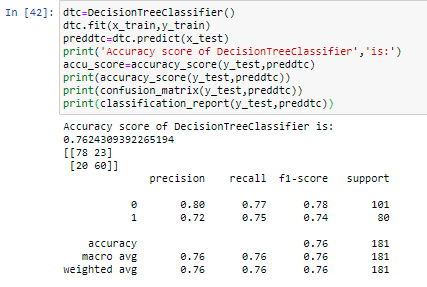
****

* We have scaled all columns in the same range of -1 to +1.
* **Classification in Machine Learning**
* Classification can be performed on structured and unstructured data as it is a technique where we categorize data into given number of classes because the main goal of classification is to identify the category/class to which a new data will fall under. In this project, we have done classification of our titanic data set using classification algorithms, like, logistic regression, k-neighbors classifier, decision tree classifier and random forest classifier.
* **Algorithms**
* **Logistic Regression**
* In Logistic Regression we are not fitting our data into a straight line instead we are mapping y v/s x to a sigmoid function. Logistic Regression uses categorical variable and the output value of logistic regression is probability of occurrence of event.
* Using the scikit learn library of python we have imported logistic regression and again fitted our model using training data and have made certain predictions in the following commands:-

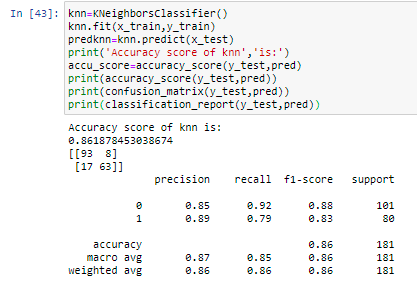




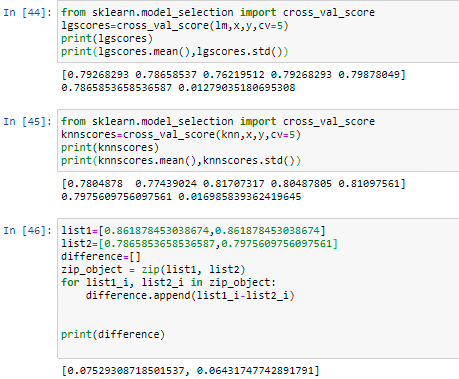
* **Decision Tree Classifier**
* Decision tree represents a function that takes input a vector of attribute values and returns a “decision”- a single output value. The input and output values can be discrete or continuous. A decision tree gives its decision by doing a sequence of tests. Decision tree has so many algorithms but the important one is ID3 (Iterative Dichotomiser) which is the most common algorithm and here dichotomization means dividing into two completely opposite things. Each attribute has divided into two most dominant attributes which is determined by calculating entropy, used, as a measure for information content of probability distribution and information gain answers us how much information an attribute provide us about the class.
* Using the Scikit learn library of python we have imported Decision Tree Classifier and again fitted our model using training data and have made certain predictions:-



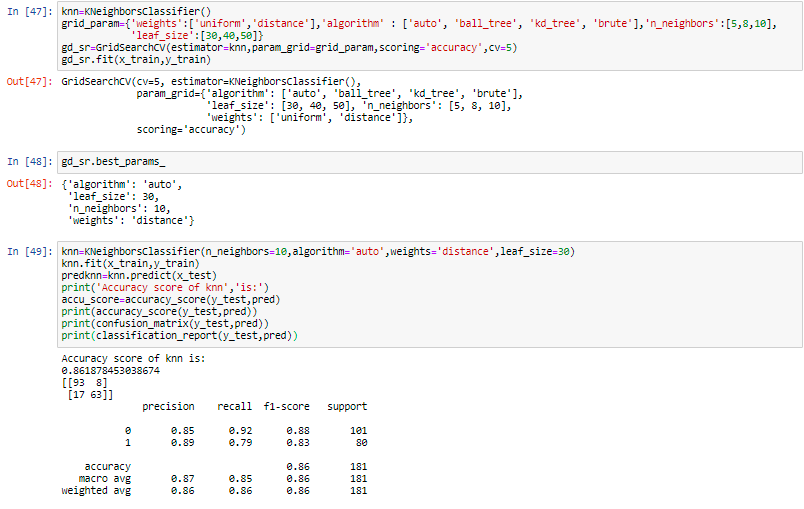
* **K-Neighbors Classifier**
* KNN or K-nearest neighbor can be used for both classification and regression predictive problems but it is widely used in classification problems in industry. In KNN, we have given a certain data based on which we have to classify the class of observed values. For example, when we have to classify on the basis of sweetness and crunchiness that whether tomato belongs to the class of fruit, vegetable or protein then we’ll use Euclidian distance algorithm to calculate minimum distance among k=3(say) nearest neighbors. In addition, then whichever class is dominant among the k-nearest neighbors becomes the class of tomato.
* Using the scikit learn library of python we have imported K Neighbors Classifier and again fitted our model using training data and have made certain predictions in the following commands :-



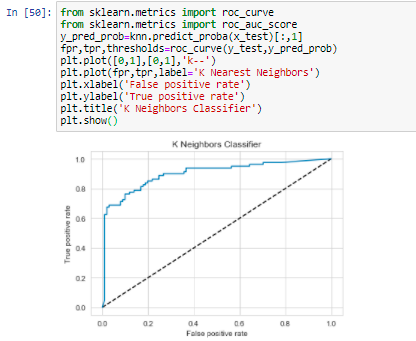
* **Concluding Remarks**
* Logistic Regression and K-Neighbors Classifier are the best algorithms.
* **Cross – Validation Scores**
* In order to find cross validation scores, we calculate cross validation scores of the two best model by using- **cross\_val\_score()** in the following command :-

****

* **Concluding Remarks**
* **K-Neighbors Classifier** is the **best algorithm** with accuracy score of **86.187 %** because the difference between accuracy score and cross validation score of k-neighbors classifier is least.
* **Hyper Parameter Tuning**
* In order to increase the accuracy score of the model, we use hyper parameter tuning of the best model in order to find best parameters by using- **GridSearchCV()** in the following commands :-

****

* **Concluding Remarks**
* After hyper parameter tuning, accuracy score of k Neighbors Classifiers has remained same.
* **AOC – ROC Curve**
* In order to verify that whether we have developed a good model or not, we form AOC – ROC Curve by using- **roc\_curve()** in the following commands :-

****

* As the line is moving towards 1, so we can conclude that we have developed good model.

**3) Data Deployment**

* In this last stage, we deploy the model to production environment so that we can make the models available to users for making predictions and important business decisions.
* In order to dump the model which we have developed so that we can use it to make predictions in future, we have saved or dumped the best model ,i.e., K-Neighbors Classifier by using- **joblib.dump()** in the following commands :-

